Escuela Superior de Ingeniería

Programación en Internet

Grado en Ingeniería Informática

Invocación de un servicio web REST desde una aplicación Android

Autores:
Javier Montes Cumbrera y Salvador Carmona Román

Supervisores:
Juan Boubeta Puig y Guadalupe Ortiz Bellot

Cádiz, 27 de mayo de 2015

Índice general

Índice de figuras

1. Software necesario

Para poder seguir este tutorial es necesario que tengamos instalado el siguiente software:

- Eclipse [1].
- SDK Android [3].
- Complemento para Eclipse [2].
- Android 4.2 y complementos.

1.1. SDK Android & Android 4.2

Para descargarnos el SDK de Android tendremos que ir hasta su página oficial [3] y hacer clic en *Download >> Installing the SDK >> Standalone SDK Tools >> Download the SDK now >> SDK Tools Only >>* Descargar el *standalone Android SDK Tools* para Windows.

Una vez tengamos descargado el *SDK* de Android lo descomprimimos, en nuestro caso será dentro de nuestra carpeta *development*, y dentro de la carpeta buscaremos la aplicación *SDK Manager*.

Después de que se abra la aplicación veremos muchos paquetes para poder instalar, tendremos que seleccionar los siguientes:

- *Android SDK Tools*
- *Android SDK Platform-tools*
- *Android SDK Build-tools 22.0.1*
- *Android 4.2.2*
- *Android Support Library*
- *Google USB Driver*

Cuando haya finalizado la descarga podremos ver algo parecido a la Figura 1.1.

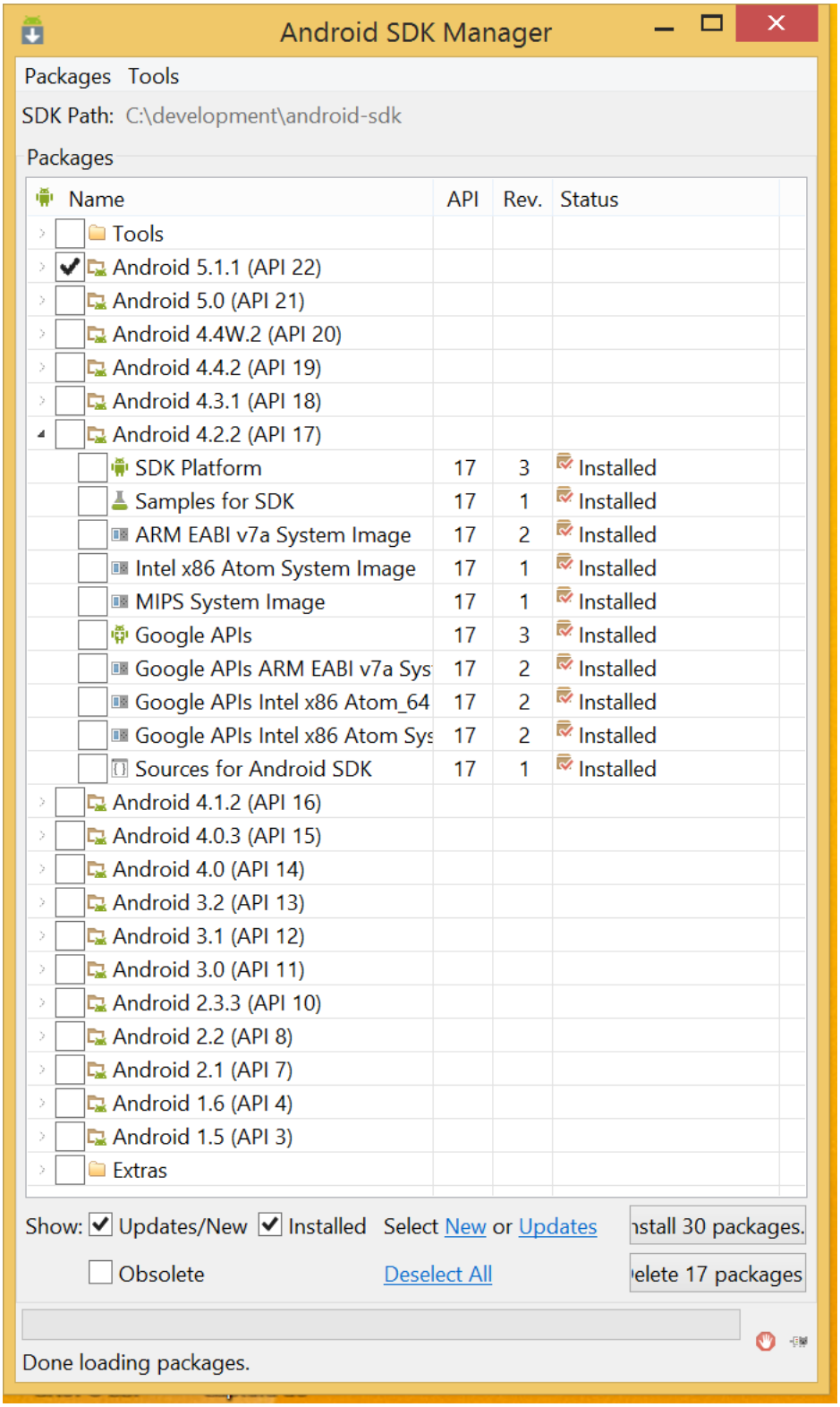

Figura 1.1.: Visión del *SDK Manager*

1.2. Complemento para Eclipse

Para poder instalar el complemento necesario para seguir el tutorial, tendremos que seleccionar la opción *Help >> Install New Software* en Eclipse. A continuación, pulsamos en el botón *Add...*, en el campo *Name* indicamos *Complemento Google Android* y en el campo *Location* la URL del plugin (*https://dl-ssl.google.com/android/eclipse*). Finalmente, seleccionamos *Developer Tools* y lo instalamos.

2. Creación e implementación del proyecto

En esta sección, crearemos un proyecto y lo dotaremos de la funcionalidad necesaria para mostrar todos los miembros del comité de programa de una conferencia almacenados en el servicio web previamente creado (véase el tutorial «Creación de un servicio Web REST y su despliegue en Tomcat»).

2.1. Creación del proyecto

Una vez instalado todo el software necesario, tenemos que crear un nuevo proyecto en Eclipse; este proyecto tendrá que ser del tipo *Android Aplication Project.* Para ello, hacer clic en *File >> New >> Other* (véase Figura 2.1).

Buscamos en la lista de opciones la categoría Android y, dentro de ella, seleccionamos *Android Aplication Project* (véase Figura 2.2).

Como nombre de la aplicación y el proyecto utilizaremos *WoEAndroid*, un nombre similar al que hemos venido utilizando hasta ahora, y como nombre del paquete utilizaremos *uca.es.woeandroid*. Además, hay que establecer la versión de Android que hayamos descargado, en nuestro caso será Android 4.2. El resultado de la configuración tendrá que ser algo parecido a lo mostrado en la Figura 2.3.

Pulsamos en siguiente hasta que nos deje elegir qué tipo de *activity* principal queremos y seleccionamos *Blank activity*. El nombre de la *activity* lo dejaremos tal y como aparece por defecto.

2.2. Añadiendo elementos a la vista principal

Antes de añadir funcionalidad a nuestra aplicación Android, tendremos que retocar el fichero *activity_main.xml* —se encuentra dentro de *res/layout*— para que tenga una lista donde mostraremos el resultado de nuestra petición al servicio. Para poder añadir el elemento lista, abriremos el fichero y nos pasaremos a la vista del código. Dentro de este código tendremos que borrar el código correspondiente a la etiqueta *TextView* y sustituirlo por el que encontramos en el Listado 2.1 (véase Anexo A).

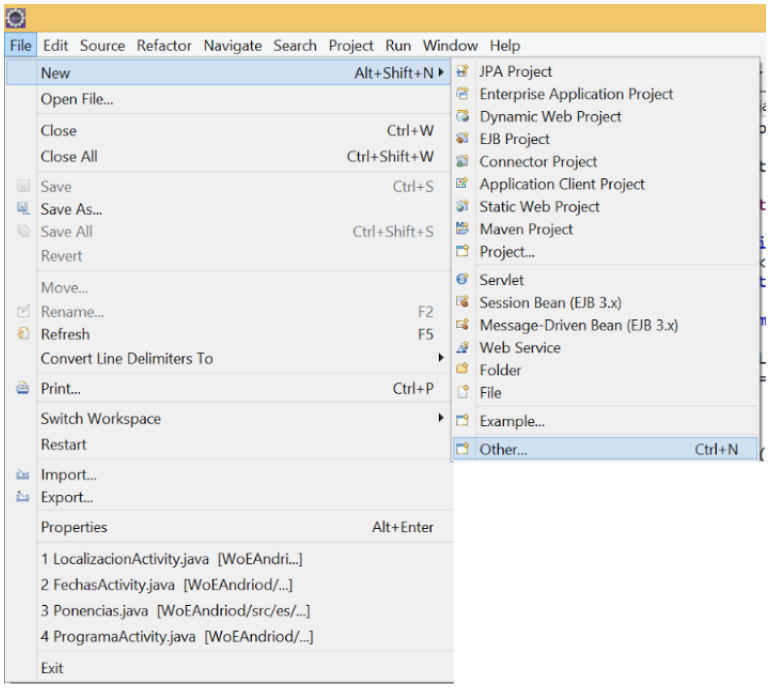

Figura 2.1.: *File >> New >> Other*

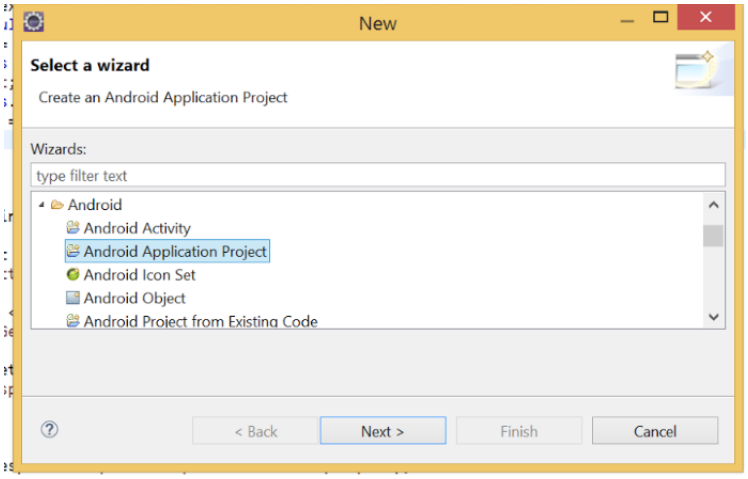

Figura 2.2.: Selección de *Android Aplication Project*

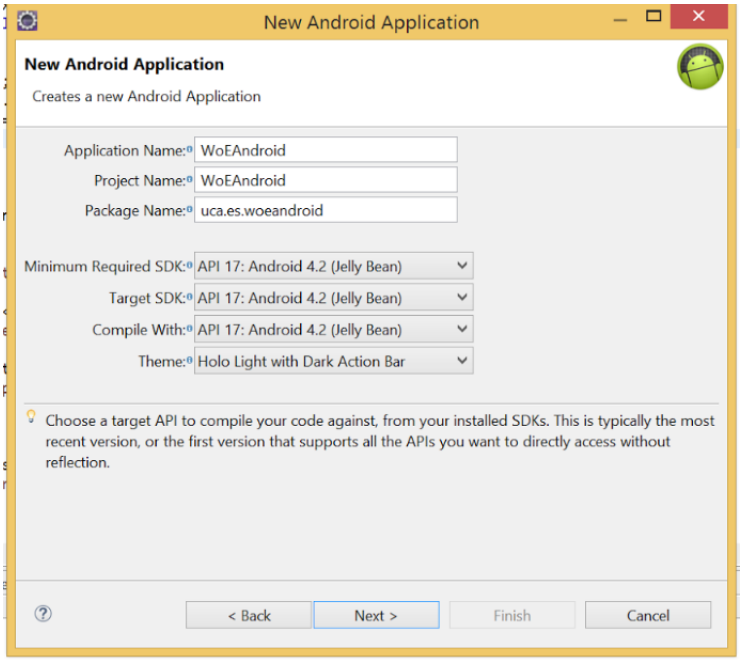

Figura 2.3.: Configuración del proyecto Android

Listado 2.1: Código para la vista principal

```
1 <ListView android:id="@+id/_listaMiembros"
2     android:layout_width="240dp"
3     android:layout_height="match_parent"
4     android:layout_gravity="start"
5     android:choiceMode="singleChoice"
6     android:divider="@android:color/transparent"
7     android:dividerHeight="0dp"
8     android:background="#111"/>
```

2.3. Creación de la clase principal

En nuestra clase principal (*MainActivity.java*) tendríamos que realizar las diferentes llamadas a nuestro servicio web, pero en Android no está permitido hacer llamadas a funciones que puedan tardar mucho o un tiempo indeterminado en responder. Para salvar este problema vamos a crear una clase externa que sea una tarea asíncrona. Así que, después de quitarle bastante peso lógico a nuestra clase principal, esta solo deberá encargarse de identificar los diferentes componentes que se encuentren en pantalla para darle funcionalidad y poder mostrar los diferentes resultados.

Lo primero que tendremos que hacer en nuestra clase principal será añadir una lista y su adaptador para poder actualizar los datos con los resultados de nuestra tarea asíncrona. Como podemos ver en el Listado 2.2, la hemos creado como variables privadas para que no sean accesibles desde el exterior (véase el código completo en el Anexo B).

Listado 2.2: Declaración de variables privadas

```
1 private ListView _mMiembros;
2 private ArrayAdapter<String> _aMiembros;
```

Dentro del método *onCreate* tendremos que enlazar la lista que acabamos de crear con la que se encuentra en la vista y también tendremos que crear una nueva instancia del adaptador. En la línea 1 del Listado 2.3 puede comprobarse que hemos utilizado la función *findViewById* para enlazar los objetos, y en la siguiente línea hemos creado una nueva instancia del adaptador indicando que será de tipo lista simple.

Listado 2.3: Inicialización de las variables privadas

```
1 _mMiembros = (ListView) findViewById(R.id._listaMiembros);
2 _aMiembros = new ArrayAdapter<String>(this,android.R.layout.
    simple_list_item_1);
```

Tras inicializar todas nuestras variables, procederemos a llamar a nuestra tarea asíncrona pasándole como parámetros nuestras variables e indicándole que ejecute su función (véase Listado 2.4).

Listado 2.4: Código para la instanciación y ejecución de la tarea asíncrona

```
1 Miembros miembros = new Miembros(getApplicationContext(),_mMiembros,
    _aMiembros);
2 miembros.execute();
```

2.4. Creación de la tarea en segundo plano

Nuestra tarea en segundo plano no tiene un fichero ya creado con nuestra *activity* principal, por lo que crearemos un fichero Java que albergue nuestra clase, denominada *Miembros* —botón derecho sobre el proyecto WoEAndroid > *New* > *Class* > *Name: Miembros*—.

Al ser una tarea asíncrona, esta tendrá que ser una extensión de *AsyncTask*, por lo que el cuerpo de nuestra clase quedaría tal y como se muestra en el Listado 2.5.

Listado 2.5: Clase Miembros recién creada

```
1 public class Miembros extends AsyncTask <Void, Void, String> {
2 }
```

Los tres parámetros que hemos especificado son necesarios para este tipo de clase, ya que identifican qué tipo de parámetros reciben los métodos *doInBrackgrund()*, *onProgressExecute()* y *onPostExecute()*. Así pues, nuestro método *doInBackground* devolverá un *String* y el método *onPostExecute* recibirá el *String*. Solamente tendremos que implementar los métodos *doInBackground* y *onPostExecute*, puesto que no se requiere sobreescribir los otros métodos.

2.4.1. *doInBackground*

En este método se ejecutará la petición a nuestro servicio web y devolverá el resultado al método *onPostExecute*.

Para poder realizar la conexión con nuestro servicio web tendremos que crear un nuevo cliente HTTP y una petición *GET*, ayudándonos de la clase *HttpGet*. A la hora de instanciar un objeto de la clase *HttpGet* tendremos que iniciarlo con la URL a la que queramos hacer la petición (véase Listado 2.6).

Listado 2.6: Variables necesarias para iniciar la petición HTTP

```
1 HttpClient httpClient = new DefaultHttpClient();
2 HttpGet httpGet = new HttpGet("http://" + _sTuIP + ":8080/WoERest/Rest/
    allMembers");
```

Como podemos observar en la línea 2 del Listado 2.6, existe una variable llamada *_sTuIP*, esta variable contendrá la IP del ordenador donde se encuentre el servicio web. En otras ocasiones hemos llamado al servicio web mediante la dirección *localhost* pero, en este caso, si utilizásemos *localhost* estaríamos haciendo referencia al propio dispositivo dando, por tanto, un fallo de conexión.

Después de tener las variables creadas, indicaremos el tipo de respuesta que estamos esperando y también recogeremos la respuesta en una variable de tipo *HttpResponse,* que posteriormente convertiremos a *String* para ser devuelta (véase Listado 2.7).

Listado 2.7: Código para la recepción de una respuesta HTTP

```
1 httpGet.setHeader("content-type", "aplication/json");
2 HttpResponse response = httpClient.execute(httpGet);
3 String respStr = EntityUtils.toString(response.getEntity());
4 return respStr;
```

2.4.2. *onPostExecute*

Una vez que tenemos la respuesta de nuestro servicio es necesario tratarla y mostrar los resultados en la vista principal. Para ello, vamos a crear un *JSONArray* que contendrá todos los JSON que estén dentro de la respuesta, lo recorreremos

añadiendo los nombres de los miembros al adaptador y, posteriormente, haremos los cambios visibles (véase Listado 2.8).

Listado 2.8: Código para mostrar los resultados

```
1 String nombre = null;
2 JSONArray a = new JSONArray(result);
3 int registros = a.length();
4 for(int i = 0; i < registros; i++) {
5     nombre = a.getJSONObject(i).getString("nombre");
6     _aMiembros.add(nombre);
7 }
8 _lListResultado.setAdapter(_aMiembros);
```

2.5. Añadiendo los permisos necesarios

Tras añadir las funcionalidades a nuestra aplicación Android podríamos probarla, pero nos daría error de conexión. Esto se debe a que no hemos configurado todavía nuestra aplicación para que pueda utilizar la conexión a Internet. Para poder conectarnos a Internet hay que añadir el código del Listado 2.9 en el fichero *AndroidManifest.xml*.

Listado 2.9: Permisos necesarios para Internet

```
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
      />
```

3. Probando la aplicación

Al finalizar todos los pasos de este tutorial, la aplicación está lista para ser probada. Pero antes tendremos que añadir un dispositivo virtual que sea capaz de ejecutar nuestra aplicación.

3.1. Creando el dispositivo virtual

Para crear el dispositivo virtual tendremos que hacer clic en el menú *Window* y seleccionar la opción *Android Virtual Device Manager* (véase Figura 3.1).

A continuación, debemos hacer clic en el botón *New* y rellenaremos la pantalla con la información que podemos ver en la Figura 3.2. Una vez que todo esté correctamente rellenado, hay que hacer clic en *Ok* y cerrar la ventana.

3.2. Ejecutando el proyecto

Para poder ejecutar la aplicación, bastará con hacer clic derecho en nuestro proyecto *WoEAndroid* y ejecutarlo como aplicación Android.

Si todo ha funcionado correctamente, en nuestra aplicación deberá haberse cargado todos los miembros del comité de programa de la conferencia que tengamos en nuestro servicio web, como se muestra en la Figura 3.3.

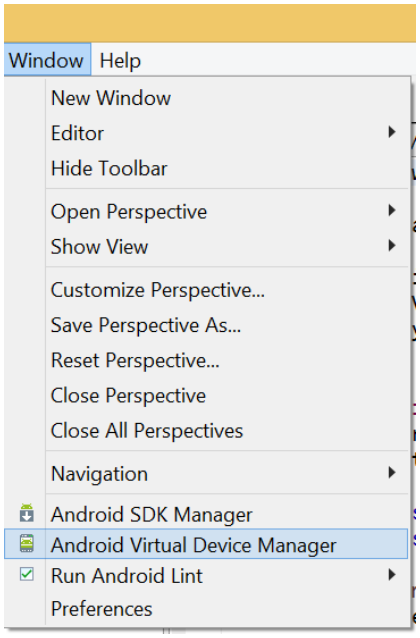

Figura 3.1.: Opción *Android Virtual Device Manager* del menú *Window*

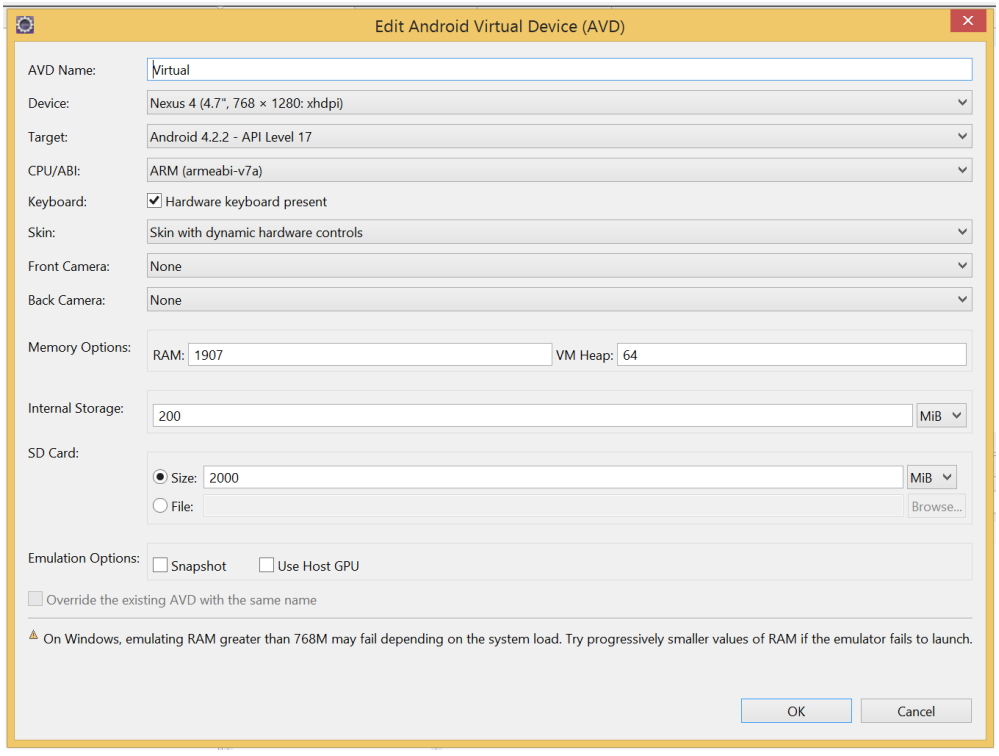

Figura 3.2.: Configuración para el *Android Device*

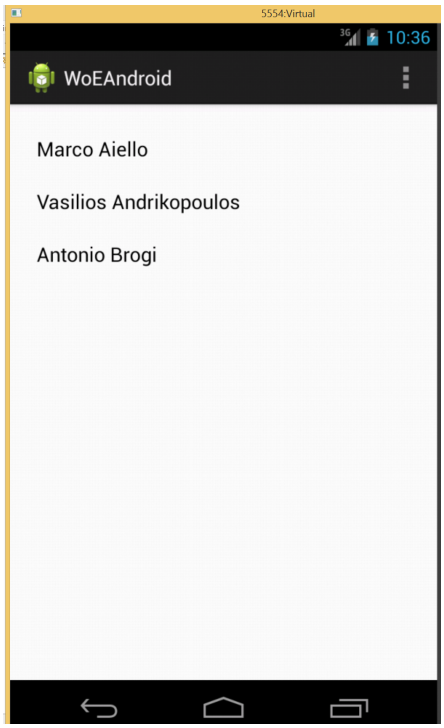

Figura 3.3.: Resultado de la ejecución de nuestra aplicación

Bibliografía

[1] Eclipse IDE for Java EE Developers | Packages (2015), https://eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/lunar

[2] Plugin Eclipse (2015), http://developer.android.com/sdk/installing/installing-adt.html

[3] SDK Android (2015), https://developer.android.com/sdk/index.html

A. *Layout activity_main*

En este anexo se encuentra el código necesario para el *layout* de la aplicación Android.

```
1  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:paddingBottom="@dimen/activity_vertical_margin"
6      android:paddingLeft="@dimen/activity_horizontal_margin"
7      android:paddingRight="@dimen/activity_horizontal_margin"
8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context="uca.es.woeandroid.MainActivity" >
10
11     <ListView android:id="@+id/_listaMiembros"
12         android:layout_width="240dp"
13         android:layout_height="match_parent"
14         android:layout_gravity="start"
15         android:choiceMode="singleChoice"
16         android:divider="@android:color/transparent"
17         android:dividerHeight="0dp" />
18
19 </RelativeLayout>
```

B. Clase *MainActivity*

En este anexo se encuentra todo el código de la clase principal de la aplicación Android.

```
1  package uca.es.woeandroid;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.Menu;
6  import android.view.MenuItem;
7  import android.widget.ArrayAdapter;
8  import android.widget.ListView;
9
10 public class MainActivity extends Activity {
11     private ListView _mMiembros;
12     private ArrayAdapter<String> _aMiembros;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18
19         _mMiembros = (ListView) findViewById(R.id._listaMiembros);
20         _aMiembros = new ArrayAdapter<String>(this,android.R.layout.
               simple_list_item_1);
21
22         Miembros miembros = new Miembros(getApplicationContext(),
               _mMiembros,_aMiembros);
23         miembros.execute();
24     }
25
26     @Override
27     public boolean onCreateOptionsMenu(Menu menu) {
28         // Inflate the menu; this adds items to the action bar if it is
               present.
29         getMenuInflater().inflate(R.menu.main, menu);
30         return true;
31     }
32
33     @Override
34     public boolean onOptionsItemSelected(MenuItem item) {
35         // Handle action bar item clicks here. The action bar will
```

```
36          // automatically handle clicks on the Home/Up button, so long
37          // as you specify a parent activity in AndroidManifest.xml.
38          int id = item.getItemId();
39          if (id == R.id.action_settings) {
40              return true;
41          }
42          return super.onOptionsItemSelected(item);
43      }
44  }
```

C. Clase Miembros

En este anexo se encuentra todo el código de la tarea asíncrona de nuestra aplicación Android.

```
1  package uca.es.woeandroid;
2
3  import org.apache.http.HttpResponse;
4  import org.apache.http.client.HttpClient;
5  import org.apache.http.client.methods.HttpGet;
6  import org.apache.http.impl.client.DefaultHttpClient;
7  import org.apache.http.util.EntityUtils;
8  import org.json.JSONArray;
9  import org.json.JSONException;
10
11 import android.content.Context;
12 import android.os.AsyncTask;
13 import android.widget.ArrayAdapter;
14 import android.widget.ListView;
15
16 public class Miembros extends AsyncTask <Void, Void, String>{
17
18     private String _sTuIp = "10.211.55.3";
19     private Context _context;
20     private ListView _lListResultado;
21     private ArrayAdapter<String> _aMiembros;
22
23     Miembros(Context context, ListView mMiembros, ArrayAdapter<String>
           aMiembros){
24         _context = context;
25         _lListResultado = mMiembros;
26         _aMiembros = aMiembros;
27     }
28
29     @Override
30     protected String doInBackground(Void... params) {
31         String respStr = null;
32         HttpClient httpClient = new DefaultHttpClient();
33         HttpGet httpGet = new HttpGet("http://"+ _sTuIp + ":8080/WoERest/
               Rest/allMembers");
34         httpGet.setHeader("content-type", "aplication/json");
35         HttpResponse response;
36         try {
```

```
37            response = httpClient.execute(httpGet);
38            respStr = EntityUtils.toString(response.getEntity());
39         } catch (Exception e) {
40            e.printStackTrace();
41         }
42         return respStr;
43      }
44
45      protected void onPostExecute(String result){
46         String nombre = null;
47         JSONArray a;
48         try {
49            a = new JSONArray(result);
50            int registros = a.length();
51            for(int i = 0; i < registros; i++){
52               nombre = a.getJSONObject(i).getString("nombre");
53               _aMiembros.add(nombre);
54            }
55         } catch (JSONException e) {
56            e.printStackTrace();
57         }
58         _lListResultado.setAdapter(_aMiembros);
59      }
60   }
```